PCI-8133

3 Channel Encoder Counter and PWM Output Card

User's Guide

Recycled Paper

© Copyright 2003 ADLINK TECHNOLOGY INC.

All Rights Reserved.

Manual Rev. 1.30: June 2, 2003

Part No: 50-11117-103

The information in this document is subject to change without prior notice in order to improve reliability, design, and function and does not represent a commitment on the part of the manufacturer.

In no event will the manufacturer be liable for direct, indirect, special, incidental, or consequential damages arising out of the use or inability to use the product or documentation, even if advised of the possibility of such damages.

This document contains proprietary information protected by copyright laws. All rights are reserved. No part of this manual may be reproduced by any mechanical, electronic, or other means in any form without prior written permission of the manufacturer.

Trademarks

NuDAQ is a registered trademark of ADLINK TECHNOLOGY INC. Other product names mentioned herein are used for identification purposes only and may be trademarks and/or registered trademarks of their respective companies.

Getting Service from ADLINK

Customer Satisfaction is the most important priority for ADLINK TECHNOLOGY INC. If you need any help or service, please contact us.

ADLINK TECHNOLOGY INC.			
Web Site	http://www.adlinktech.com		
Sales & Service	Service@adlinktech.com		
Technical Support	NuDAQ + USBDAQ + PXI	nudaq@adlinktech.com	
	Automation	automation@adlinktech.com	
	NuIPC	nuipc@adlinktech.com	
	NuPRO / EBC	nupro@adlinktech.com	
TEL	+886-2-82265877	FAX	+886-2-82265717
Address	9F, No. 166, Jian Yi Road, Chungho City, Taipei, 235 Taiwan.		

Please email or FAX us your detailed information for prompt, satisfactory, and consistent service.

Detailed Company Information			
Company/Organization			
Contact Person			
E-mail Address			
Address			
Country			
TEL		FAX	
Web Site			
Questions			
Product Model			
Environment	OS: Computer Brand: MB: CPU: Chipset: BIOS: Video Card: NIC: Other:		
Detailed Description			
Suggestions for ADLINK			

Table of Contents

Chapter Outline ..... **iv**

Chapter 1 Introduction **1**
1.1 Features 2
1.2 Applications 2
1.3 Specifications 3
1.4 Supported Software 5

Chapter 2 Installation **6**
2.1 Package Contents 7
2.2 Unpacking 7
2.3 PCI-8133's Layout 8
2.4 Hardware Installation Outline 9
2.5 Connector Pin Assignment 10
2.6 Signal Connection 12
2.6.1 Encoder Input Circuit 12
2.6.2 General Purpose Differential D/I signals 12
2.6.3 PWM Output Circuit 13
2.6.4 Isolated Digital Outputs 13
2.6.5 Isolation Digital Inputs 14
2.7 Daughter Board Connection 14
2.7.1 Connect with ACLD-9137 14
2.7.2 Connect with ACLD-9188 14

Chapter 3 Operation Theory **15**
3.1 Encoder Counters 15
3.1.1 Differential Input & Isolation 16
3.1.2 3-Stage Digital Filter 16
3.1.3 Quadrature Decoder 17
3.1.4 Position Counter and Data Latch 18
3.1.5 Special Counter Operation Mode 18
3.2 Programmable Interrupt Counter 19
3.3 Index Latch Register 20
3.4 PWM Signal Generator 20
3.5 Interrupt Control 21
3.5.1 System Architecture 21
3.5.2 INT0 Timer 22
3.5.3 INT1 Timer 22

3.5.4. IRQ Level Setting.....22
3.5.5 Dual Interrupt System..... 23

Chapter 4 Registers..... 24
4.1 I/O Port Address 24
4.2 Counter Registers..... 25
4.3 Index Registers..... 26
4.4 Status Register 27
4.5 Digital Output Register..... 27
4.6 Control Mode Register 28
4.7 Interrupt #0 Period Register..... 30
4.8 Interrupt #1 Period Register..... 30
4.9 PWM Output Registers 31
4.10 Digital Input Register..... 31
4.11 Interrupt Clear Registers..... 32

Chapter 5 C/C++ Library..... 33
5.1 Installation..... 33
5.1.1 MS-DOS Software Installation..... 33
5.1.2 Device Installation for Windows 95/98/NT/2000..... 33
5.2 C/C++ Programming Library 34
5.3 _8133_Initial..... 35
5.4 _8133_Software_Reset..... 36
5.5 _8133_Read_Cnt..... 37
5.6 _8133_Read_Index..... 38
5.7 _8133_Read_Status 39
5.8 _8133_CLR_IdxLah 40
5.9 _8133_ModeSelecct 41
5.10 _8133_Set_Int0Perd 42
5.11 _8133_Set_Int1Perd 43
5.12 _8133_Set_PWMPerd 44
5.13 _8133_INT_Control..... 45
5.14 _8133_CLR_IRQ0 & _8133_CLR_IRQ1..... 46
5.15 _8133_Get_IRQ_Channel..... 47
5.16 _8133_Get_IRQ_Status..... 48
5.17 _8133_DO..... 49
5.18 _8133_DI 50
5.19 _8133_INT_Enable 51
5.20 _8133_INT_Disable 52

Appendix A DOS Example Programs..... 53

Appendix B PWM Example 55

Appendix C PWM Duty Cycle Example 57

Warranty Policy 61

Chapter Outline

This manual is designed to help you use the PCI-8133. The manual describes how to modify various settings on the PCI-8133 card to meet your requirements. It is divided into five chapters:

Chapter 1 Introduction

Provides an overview of the product features, applications, and specifications.

Chapter 2 Installation

Describes how to install the PCI-8133.

Chapter 3 Operation Theory

Details operations of the PCI-8133.

Chapter 4 Registers

Describes the register structure of the PCI-8133 for low-level programming.

Chapter 5 C/C++ Function Library

Details high-level programming in C/C++.

1

Introduction

The PCI-8133 is a 3-channel quadrature encoder counter card for the 32-bit PCI bus. This card is suitable for motor control and/or position monitoring for optical-mechanical systems. Features of the PCI-8133 includes three 16-bit quadruple AB phase encoder counters, three 12-bit PWM signal outputs, and general purpose isolated digital input and output channels.

Each encoder counter is equipped with digital de-glitch filters and an on-board 5000Vrms isolation circuit. The multi-configuration abilities of the input signals allow users to apply the card to various motion control applications.

1.1 Features

The PCI-8133 PCI Bus Advanced Data Acquisition Card provides the following advanced features:

- 32-bit PCI Bus, Plug and Play
- Three quadruple AB phase encoder counters
- 16-bit up/down counters
- Digital de-glitch filters for each encoder input signal
- Programmable digital de-glitch filter frequency
- On-board 5000Vrms photo-isolation circuit for encoder and digital I/O signals
- Three 12-bit PWM waveform generators
- Dual interrupts from two programmable timer clock signals
- Compact in size – half-sized PCB
- One 37-pin rugged D-type connector for encoder signals
- One 40-pin header connector for digital I/O expansion

1.2 Applications

- Motion control
- Process monitoring
- Industrial process control

1.3 Specifications

Encoder Counter Input

- Number of channels: 3
- Counter resolution: 16-bit
- Encoder counters:
  - Up/down counter
  - Digital filter for input signals
  - A phase, B phase, and index inputs
- Counter 1 input signals:
  - A phase and B phase decoder inputs
  - VCO (CCW+CW pulse) input
- Counter 2 input signals:
  - A phase and B phase decoder inputs
  - Pulse command input
- Counter status readback
- Filter type: 3-order digital filter
- Digital filter frequency: programmable 10MHz, 5MHz, 2.5MHz, 625kHz

PWM Signal Output

- Number of channels: 3
- Signal resolution: 12-bit
- Base frequency: 10MHz
- PWM cycle is synchronized with interrupt signal

Isolation Digital Output/Input

- Number of input channels: 11
- Number of output channels: 8
- Input voltage: 0-24Vdc
  - Logical H: 3~24V
  - Logical L: 0~1.5V
- Input resistance: 1.2kΩ @ 0.5W
- Digital output type: Darlington Transistors, open collector up to 40Vdc
- Sink current: 345mA typical, 500mA maximum per channel
- Isolated voltage: 2500Vrms

General Specifications

- Connectors: one 37-pin D-type female and one 40-pin header connector
- Interrupt sources (Dual Interrupt):
  - Internal Timer Clock 1
  - Internal Timer Clock 2
- Operating temperature: 0° ~ 55°C
- Storage temperature: -20° ~ 80°C
- Humidity: 0~95%, non-condensing
- Dimensions: 162mm x 105mm
- Power Requirement: +5V @ 580mA (typical)

1.4 Supported Software

ADLINK provides versatile software drivers and packages for users' differing approaches to building a system. Programming libraries, such as DLLs for most Windows based systems, are included. All software options are located in the ADLINK CD.

Programming Library

For customers who write their own programs, we provide function libraries for many different operating systems, including:

- **DOS:** Borland C/C++ function descriptions are included in this user's guide.
- **Windows 95/98/NT/2000**: VB, VC++, Delphi, and BC5, function descriptions are included in this user's guide
- **Linux:** Device drivers for the PCI-8133 are compiled as kernel modules. The binary modules have been tested under kernels 2.2.12, 2.2.14, and 2.2.16.

  ADLINK provides a TGZ file for the Linux kernel and related GLIBC. The TGZ files are placed under the `/Motion Control/PCI-8133/Linux` directory of the CD. The following are included:

  - PCI8133.tgz:
    - Binary compatible with kernels 2.2.12, 2.2.14, and 2.2.16
    - Tested with GLIBC 2.1.3, gcc 2.9.1.66 and Perl 5.005

2

Installation

This chapter describes how to install the PCI-8133. The following sections are covered in this chapter.

- Package Contents (section 2.1)
- Unpacking (section 2.2)
- PCI-8133 Layout (section 2.3)
- Hardware Installation Outline (section 2.4)
- Connector Pin Assignment (section 2.5)
- Signal Connection (section 2.6)
- Daughter Board Connection (section 2.7)

The PCI-8133 automatically configures the IRQ, port and BIOS addresses. Therefore, it is not necessary to configure these addresses, hence avoiding addressing conflicts

2.1 Package Contents

In addition to this *User's Manual*, the also package includes the following items:

- PCI-8133 Enhanced Multi-function Data Acquisition Card
- 40-pin to 37-pin connector with bracket
- ADLINK All-in-one CD
- DIN-37D, ACLD-9137, ACLD-9188, and Terminal Board (Optional)

If any of these items are missing or damaged, contact the dealer from whom you purchased the product. Save the shipping materials and carton in case you need to ship or store the product in the future.

2.2 Unpacking

The card contains electro-static sensitive components that can be easily damaged by static electricity.

Therefore, the card should be handled on a grounded anti-static mat. The operator should be wearing an anti-static wristband, grounded at the same point as the anti-static mat.

Inspect the card module carton for obvious damage. Shipping and handling may cause damage to your module. Be sure there is no shipping and handling damage on the carton before continuing.

After opening the carton, remove the system module and place it only on a grounded anti-static surface with the component side up.

Again, inspect the module for damage. Press down on all the socketed ICs to make sure that they are properly seated. Do this only with the module placed on a firm flat surface.

Note: DO NOT ATTEMPT TO INSTALL A DAMAGED BOARD IN THE COMPUTER.

You are now ready to install your card

2.3 PCI-8133's Layout

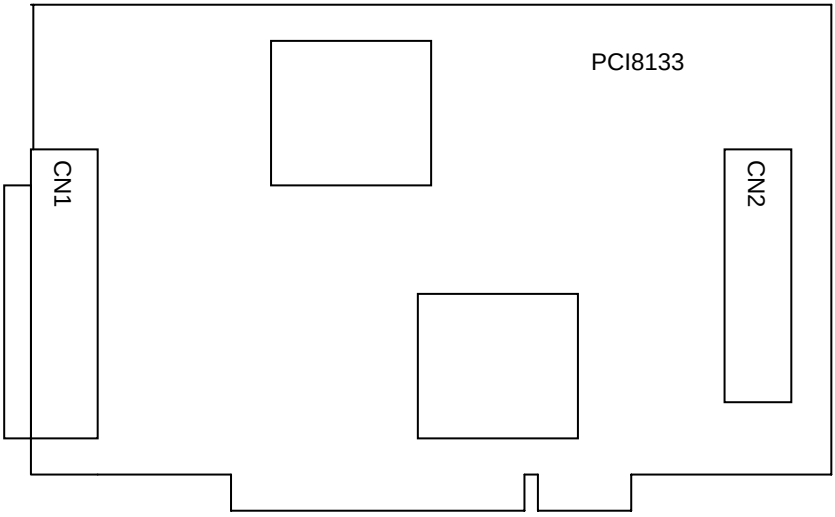


Figure 1. PCB Layout of the PCI-8133

2.4 Hardware Installation Outline

PCI configuration

The PCI card (or CompactPCI card) is equipped with Plug and Play PCI controllers. It can request base addresses and interrupts according to PCI standards. The system BIOS will install the system resources based on the PCI cards' configuration registers and system parameters (also set by the system BIOS). Interrupt assignment and memory usage (I/O port locations) of the PCI cards can only be assigned by system BIOS. These system resource assignments are done on a board-by-board basis. It is not suggested to assign the system resource by any other methods.

PCI slot selection

The PCI card can be inserted into any PCI slot without any configuration modification to the system resources. Please note that the PCI system board and slot must provide bus-mastering capabilities to operate at optimum level.

Installation Procedures

1. Turn off your computer.
2. Turn off all accessories (printer, modem, monitor, etc.) connected to your computer.
3. Remove the cover from your computer.
4. Set the jumpers on the PCI or CompactPCI card.
5. Select a 32-bit PCI slot. PCI slots are shorter than ISA or EISA slots, and are usually white or ivory.
6. Before handling the PCI card, discharge any static buildup on your body by touching the metal case of the computer. Hold the edge of the card and do not touch the components.
7. Position the board into the PCI slot you have selected.
8. Secure the card in place at the rear panel of the system.

2.5 Connector Pin Assignment

The PCI-8133 comes equipped with one 37-pin D-type connector (CN1) and one 40-pin pin header (CN2). CN2 can be converted to a 37-pin D-type connector. The pin assignment for CN1 and CN2 are illustrated in Figure 2. Please ensure that the 40-pin to 37-pin connector is included in the package.

- CN 1: Encoder Input Signals & PWM Output
- CN 2: Isolation Digital Input & Output

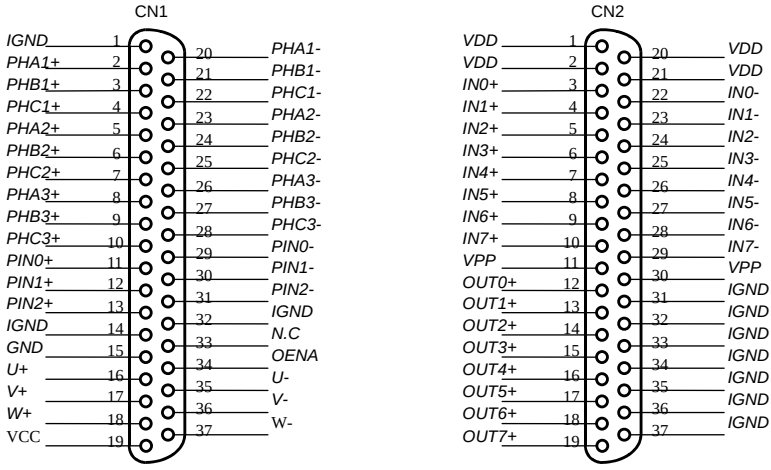


Figure 2. Pin Assignment of CN1 & CN2

Note: CN2 shown here has been converted from 40-pin header to 37-pin DSUB connector

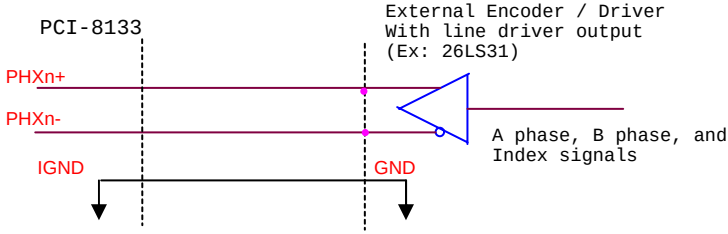
Legend:

PHXn+:	Positive arm of the differential photo encoder input
PHXn-:	Negative arm of the differential photo encoder input [X=A, B, C, and n=1~3]
PINk+:	General purpose differential D/I signals
PINk-:	General purpose differential D/I signals [k=0~2]
IGND:	Isolated Signal Ground referenced to VDD
VDD:	Isolated Voltage Output from bus +5V
OENA:	PWM signal output enabled
VCC:	Bus voltage output +5V
GND:	Bus ground correspond to VCC for PWM signal
U+/-:	PWM Output, U channel positive / negative
V+/-:	PWM Output, V channel positive / negative
W+/-:	PWM Output, W channel positive / negative
INm+/-:	Differential Digital Input CH-m positive / negative
OUTm:	Isolated Digital Output CH-m
VPP:	Fly wheel power line input for Isolation digital output

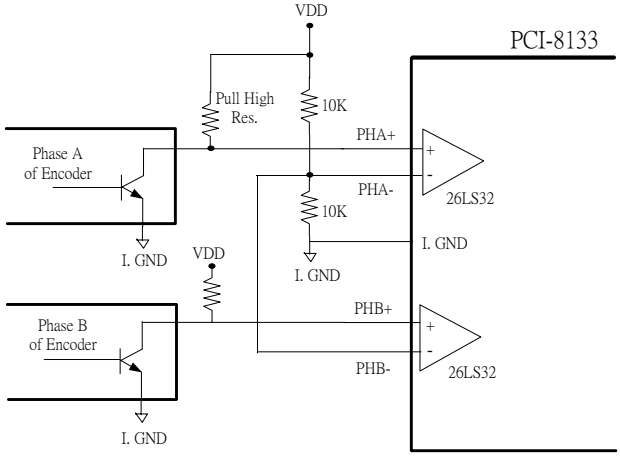
2.6 Signal Connection

2.6.1 Encoder Input Circuit

Encoder output signals for differential



Encoder output signals for single-ended

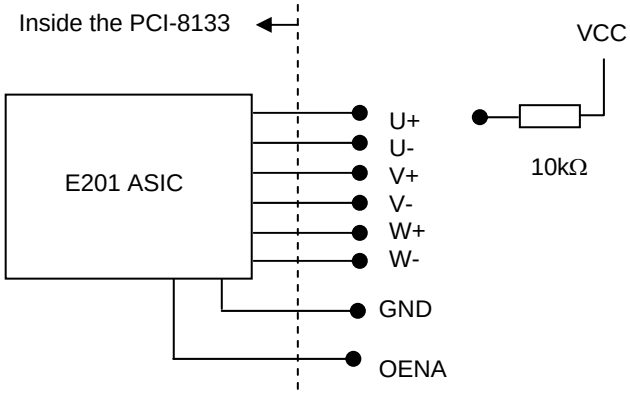


2.6.2 General Purpose Differential D/I signals

There are 3 general-purpose differential isolated D/I signals available to CN1. The input circuit of these signals are the same as the encoder input signals. They are differential input with 220 Ohms resistive loading. The signals and ground have a 5000Vrms isolation voltage between the PC ground and power.

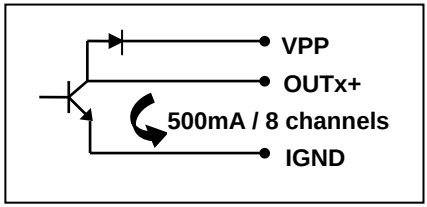
2.6.3 PWM Output Circuit

The PWM is an open collector output. The max sink current is 20mA. Users can cascade a 10kΩ resister with VCC to test it.



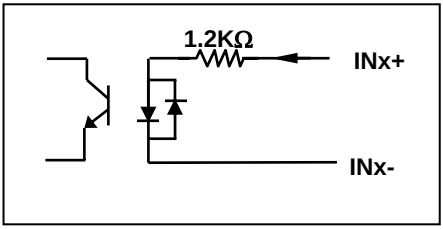
2.6.4 Isolated Digital Outputs

The isolated digital output is an open collector transistor output. The connection of the isolated digital output is shown in the diagram below. When the isolated digital output goes high, the sink current will be from the DO channel n.



2.6.5 Isolation Digital Inputs

The isolation digital input accepts voltages between 0V and 24V with an input resistance of 1.2KΩ. The connection between the outside signal and the PCI-8133 is shown in the illustration below. Please note that there is no reference ground for a digital input channel. The DI channels are isolated from other channels.



2.7 Daughter Board Connection

The PCI-8133 can be connected with several different daughter boards, including the ACLD-9137 and ACLD-9188. The functionality and connections are described below.

2.7.1 Connect with ACLD-9137

The ACLD-9137 is a connector for cards, which are equipped with 37-pin D-sub connector. The ACLD-9137 board provides an efficient way to connect for simple applications that do not need complex signal conditioning before an A/D conversion is performed.

2.7.2 Connect with ACLD-9188

The ACLD-9188 is a general-purpose terminal board for all cards, which comes equipped with a 37-pin D-sub connector.

3

Operation Theory

This chapter describes, in detail, the operation of the PCI-8133 card. Contents covered include:

- Encoder Counter Input
- Programmable Interrupt Counter
- Index Latch Register
- PWM Generator
- Interrupt Control

3.1 Encoder Counters

There are 3 independent 16-bit up/down counters, which are typically used for speed and position monitoring of a motion control system. These counters, preceded by digital filter circuits, can count pulses from A/B phase encoders or other types of position sensors. The flowchart of signal processing is shown in Figure 3.

Counter 3 is dedicated for A/B phase signal input. However, Counter 1 and Counter 2 are capable of running in different operation modes such as Pulse/Direction input mode. Note that the digital filters are only available for the A/B phase input mode.

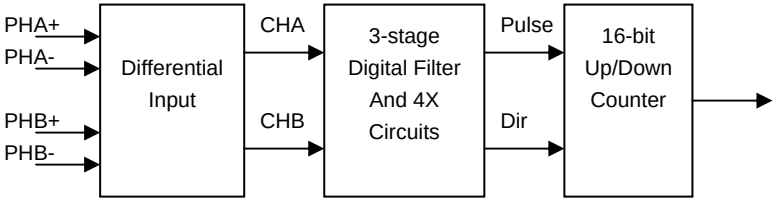


Figure 3. Encoder Interface Block Diagram

3.1.1 Differential Input & Isolation

The encoder input signals are in differential pairs. Differential signals are transformed into single-ended signals by a differential driver. The loading for each differential input signal is 220 Ohms. Single-ended signals are isolated from the host power and ground by a photo-coupler, which as an isolation voltage of 5000Vrms.

3.1.2 3-Stage Digital Filter

The output signal from the photo-couplers passes through a 3-stage digital filter. This circuit can filter out noise spikes that typically occur in motor system application. Utilizing the filters, users can improve the accuracy of the system.

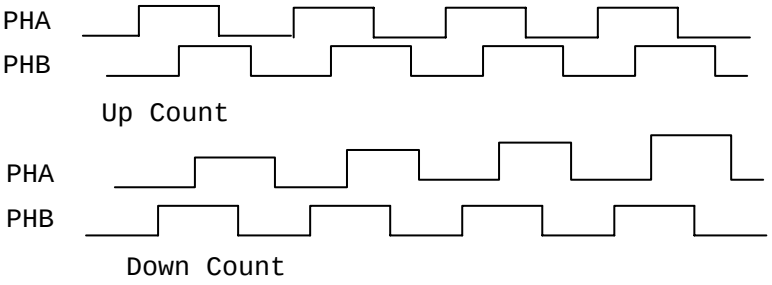
Signals from each channel are sampled on the rising clock edge. A time history of the signal is stored in a four-bit shift register. Any change on the input is tested for a stable level for three consecutive rising clock edges. Therefore, the filtered output waveform can change only after an input level has the same value for three consecutive rising clock edges, thus filtering out any incoming noise.

The sampling period of the filters can be set by software. Two bits (FTS1 and FTS0) are used to select the operation frequency. Refer to section 4.6 for the definition of these bits.

3.1.3 Quadrature Decoder

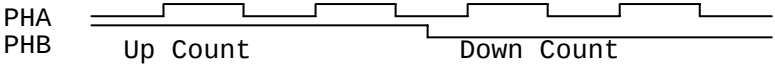
The quadrature decoder decodes the incoming filtered signals into pulses for counting. The circuitry multiplies the position resolution of the input signals by a factor of four (4X decoding). When using an encoder for motor position sensing, the increased resolution can provide precise system control. For example, for an A/B phase encoder with 2000 pulses per revolution, a resolution of 8000 pulses per revolution can be achieved.

The quadrature decoder samples the outputs from CHA and CHB filters. It outputs a pulse signal and a direction signal to the internal position counter based on the previous binary state and present state of the two signals.

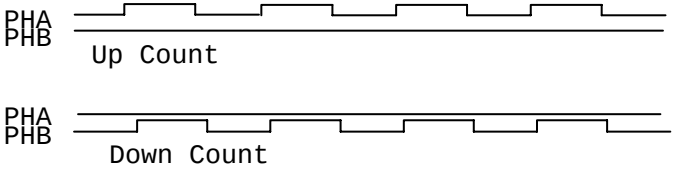


Quadrature Mode (A/B Phase Mode)

Pulse/Direction Mode (OUT/DIR Mode)



Up/down counter Mode (CW/CCW Mode)



3.1.4 Position Counter and Data Latch

This 16-bit binary up/down counters count on the rising clock edges. The 16 bits of data are passed to the position data latch after the counter value changes. Counter values are 0 to 65535. When the position (counter) range of a system exceeds 65535, user should use an interrupt service routine (ISR) to carry out a position-monitoring task to avoid any losses in position information. ISR techniques are explained in section 3.5.

The position data latch is a 16-bit latch that captures the position counter output data on each rising clock edge, except when its inputs are disabled by the inhibit logic section during data read operation. The output data is passed to the local data bus. When active, a signal from the inhibit logic section prevents new data from being captured by the latch, keeping the data stable while the read operation is made through the bus interface. The latch is automatically re-enabled at the end of the read operation. The latch is cleared to 0 asynchronously by the _8133_Software_Reset() function.

The counters' value cannot be set directly. It can only be cleared to 0 using the _8133_Software_Reset() function

3.1.5 Special Counter Operation Mode

The PCI-8133 card can also accept up/down pulse or pulse/direction type signal inputs depending on the sensors used. The following table shows the two types of pulse signals. Counter 1 can accept two types of signal inputs and Counter 2 can accept four types. The _8133_ModeSelect() function is used to set these modes. Bit 1 MS is for Counter 1 and (C2M1, C2M0) are for Counter 2. Refer to Section 4.6 for details of the settings.

	OUT/DIR	**CW/CCW**	**A/B**	**Filter**
Counter 1	No	Yes	Yes	No
Counter 2	Yes	Yes	Yes	Yes
Counter 3	No	No	Yes	No

3.2 Programmable Interrupt Counter

There are two programmable interrupt sources, INT0 and INT1 supported by the PCI-8133 card. The interrupt signals can help with calculating motor speed or/and monitor motor position.

Users can obtain position information from the encoder every time an interrupt is generated, for example PLS(n) is the pulse read at the n-th sampling time. PLS (n-1) is the pulse at the (n-1)th sampling time. Thus, to obtain speed information, subtract these two values:

VEL(n)=K[PLS(n)-PLS(n-1)],

Where K represents the unit transform coefficient. Let

dPLS(n)=PLS(n)-PLS(n-1),

If we integrate this information with the ISR, then the position information can be obtained with an unlimited range. That is,

POS(n)=POS(n-1)+dPLS(n),

Where POS(n) represents the position information at n-th sampling time. A simple example program is available in the Appendix.

Please ensure a suitable interrupting period is chosen. The example in the Appendix shows how to set the interrupting period. The principle is not to allow dPLS(n) exceed ±65536 (the maximum counter range). Users must calculate the maximum input pulse frequency and choose a suitable interrupting period.

For example, if an encoder with 5000 pulses per round for each A/B phase and the maximum velocity of this motor is 3000 rpm (revolution per minute), the maximum input pulse frequency will be F_max.

F_max = 5000 x 4 x (3000/60) = 1000000(Hz).

If the period chosen for INT0 is 1ms, then dPLS(n) = 1000. This will be the maximum pulse difference between the two interrupts. The shorter the interrupt period set, the better the dynamic response for sensing, but resolution and CPU run time are reduced. The period of INT0 can be set with the _8133_Set_INT0Perd() function. The possible range is from 0.1us to 6.5ms. The period of INT1 can be set with the _8133_Set_INT1Perd() function. The period of INT1 must be n times INT0's ($0 < n < 256$). Using this method, the range of INT1 can be extended from 0.1us to about 1.7s.

These two interrupts can be accepted by the host CPU only when they are enabled individually with the _8133_Set_Int_Control() function. When entering the ISR, _8133_CLR_IRQ0() or _8133_CLR_IRQ1() functions must be executed to clear the interrupt requests to the host CPU.

3.3 Index Latch Register

The PCI-8133 card provides three index latch registers. Data from the counter is latched into the register on the rising edge of each corresponding index signal (PHCx, x=1, 2, 3). These registers can be read by the _8133_Read_Index() function. At the same time, the latching status of the register will be recorded in the status register and can be read by the _8133_Read_Status() function.

Three bits (IDL1, IDL2, IDL3) are used to show the status. They are all zeroed out when reset, and become 1 when a rising edge of the index signal from PHCx is detected, where IDLx corresponds to the channel index signal from PHCx. Users can check if an index position is reached by polling the status of these three bits continuously. IDLx will be 1 once an index signal is reached, thus users will have to clear IDLx using the _8133_CLR_IdxLah() function before the next index signal enters. This function is beneficial to users performing the "Origin Return" function.

3.4 PWM Signal Generator

This function is used to generate 3 complementary PWM signals (UU, UD), (VU, VD), (WU, WD). These PWM signals can be used either for 3 phase power transistor control or used as a simple Digital to Analog converter with a low pass filter. Note that the output of these six PWM signals is an open collector, and they can sink current up to 20mA so they can interface with photo-couplers directly. External pull up resistor is necessary when used as simple D/A converter.

The carrier frequency of PWM signals is synchronized with INT0, and frequency is designed to be half of INT0's frequency. For example, if the INT0 frequency is 10 kHz, then the frequency of the PWM signals are automatically 5kHz.

The length of each PWM signal can be set by the _8133_PWMPerd() function and must be less than the INT0 period. For example, if the value set by _8133_Set_INT0Perd() is 1000 for a 10kHz frequency, and the value in _8133_set_PWMPerd() function is 500, then the on duration of one INT0 cycle is 100us with the duration of one PWM cycle being 200us (5KHz).

Users can write in a value for the PWM period at any time. The on duration is changed at the next INT0 period. When used in a power transistor control application, a dead time is necessary to prevent a short circuit between the upper and lower transistors (i.e. UU and UD). A Dead Time Generator circuit is provided with the _8133_SET_DT() function to setup dead time length. Refer to Appendix C for the relationship between PWM signals, INT0, and dead time.

The lower byte of the register value is used to set the dead time of 3 PWM complementary signals (UU, UD), (VU, VD), and (WU, WD). The formula to set this period is as follows:

TDT = 0.75 * (m + 1); (us) (0<m<256)

Where TDT is the physical dead time generated in microsecond units.

Refer to section 5.11 *_8133_set_IntPerd()* and section 4.8 Interrupt #1 period register bit N0~N7 for dead time settings.

3.5 Interrupt Control

3.5.1 System Architecture

The PCI-8133's interrupt architecture is a powerful and flexible ***Dual Interrupt System*** that is suitable for motion control applications. "Dual Interrupt" means that the hardware can generate two interrupt request signals at the same time, and that the software can service these two request signals by ISR. Note that the dual interrupt does not mean the card occupies two IRQ levels.

The two interrupt request signals (INT0 and INT1) come from the onboard timers.

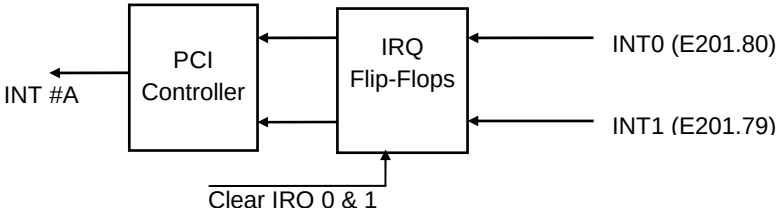


Figure 4. Dual Interrupt System of PCI-8133

3.5.2 INT0 Timer

The INT0 Timer can generate INT0 interrupt signal under a 10MHz timer base, which is 12 bits.

When the PCM bit is default 0, the formula for setting the interrupt period of INT0 is as following:

TINT0 = 0.2 (us) x Timer Value (0 < Timer Value < 4096)

When the PCM bit is set to 1, the formula for setting the interrupt period of INT0 is as following:

TINT0 = 0.1 (us) x Timer Value (0 < Timer Value < 4096)

Note: The PCM bit is in Control Register bit 6

3.5.3 INT1 Timer

INT1 Timer generates the INT1 interrupt signal. The period of INT1 is (n+1) times the INT0 period (0 < n < 256). Using INT1, the interrupting period can be adjusted to as long as needed. This is an 8-bit timer. The formula for setting the INT1 interrupt period is:

TINT1 = TINT0 * (n+1) (us); (0<n<256)

3.5.4. IRQ Level Setting

There is only one IRQ level available to the card, although it is a dual interrupt system. This card uses INT #A interrupt request signal on the PCI bus. The motherboard circuits will transfer INT #A to one of the AT bus IRQ levels. The IRQ level is set by the PCI plug and play BIOS and is saved in the PCI controller. It is not necessary for users to set the IRQ level.

3.5.5 Dual Interrupt System

The PCI controller of PCI-8133 can receive two hardware IRQ sources. However, a PCI controller can generate only one IRQ to the PCI bus. The two IRQ sources should be distinguished by the ISR of the application software if two IRQs are all used.

The application software can use the "_8133_Get_Irq_Status" function to distinguish which interrupt is inserted. After servicing an IRQ signal, users should check if another IRQ is also asserted and then clear the current IRQ to allow the next IRQ. The two IRQs are INT0 and INT1 that come from the interrupt generator.

Note that even if you disable both the IRQ sources without changing the initial condition of the PCI controller, the PCI BIOS will still assign an IRQ level to the PCI card and it will occupy the PC resource. It is not suggested to redesign the initial condition of the PCI card by users' own application software. If users want to disable the IRQ level, please use the ADLINK software utility to change the power on the interrupt settings.

4

Registers

The description of the registers and structure of the PCI-8133 are outlined in this chapter. The information in this chapter will assist programmers develop low-level programs for the card. However, we strongly recommend the use of the standard drivers in the ADLINK CD.

4.1 I/O Port Address

The PCI-8133 functions as a 32-bit PCI target device to any master on the PCI bus. There are three types of registers on the PCI-8133: PCI Configuration Registers (PCR), Local Configuration Registers (LCR), and PCI-8133 registers.

The PCRs, which are PCI-bus compliant, are initialized and controlled by the system's Plug and Play PCI BIOS. Users can study the PCI BIOS specifications to understand the operation of the PCR. The PCR can only be read by the PCI BIOS function call.

The LCRs are specified by the PCI bus controller, PCI-9052, from PLX Technology Inc. (www.plxtech.com). It is not necessary for users to understand the details of the LCR if you use the software library we provided. The base address of the LCR is assigned by the PCI Plug and Play BIOS. The assigned address is located at offset 14h of PCR.

The PCI-8133 registers are shown in the Table 4.1. The base addresses of the PCI-8133 registers are also assigned by the PCI's Plug and Play BIOS. The assigned base address is located at offset 18h of the PCR. All the PCI-8133 registers are 16 bits. The users can access these registers by 16-bit I/O instructions.

Users can read the PCR to get the LCR base address and the PCI-8133 base address by using the PCI BIOS function call.

I/O Base Address	Write	Read
Base + 00h	--	16-bit Counter Value 1
Base + 02h	--	16-bit Counter Value 2
Base + 04h	--	16-bit Counter Value 3
Base + 06h	Clear Index Register 1	Counter Index for Counter 1
Base + 08h	Clear Index Register 2	Counter Index for Counter 2
Base + 0Ah	Clear Index Register 3	Counter Index for Counter 3
Base + 0Ch	--	Status Register
Base + 10h	Digital Output Register	--
Base + 12h	Control Register	--
Base + 14h	Reserved	--
Base + 16h	16-bit INT0 Period Register	--
Base + 18h	8-bit INT1 Register	--
Base + 1Ah	12-bit PWM U Channel	--
Base + 1Ch	12-bit PWM V Channel	--
Base + 1Eh	12-bit PWM W Channel	--
Base + 40h	--	Digital Input Channels
Base + 80h	Clear H/W INT0	--
Base + 90h	Clear H/W INT1	--

Table 1. I/O Address MAP

4.2 Counter Registers

There are 3 independent unsigned 16-bit up/down counters. The counter register stores the value of the counter. Please refer Section 3.1 for details of the counter operation.

Address: BASE + 0x00h / BASE + 0x02h / BASE + 0x04h

Attribute: Read only

Data Format:

Bit	7	6	5	4	3	2	1	0
BASE+0/2/4	CB7	CB6	CB5	CB4	CB3	CB2	CB1	CB0
BASE+1/3/5	CB15	CB14	CB13	CB12	CB11	CB10	CB9	CB8

CB15…CB0: Counter value.

CB15 is the Most Significant Bit (MSB),

CB0 is the Least Significant Bit (LSB).

4.3 Index Registers

When the PHCn signal is triggered, the index register is used to store the counter value and set the Index Latch (IDLn) bit in the status register. Refer to Section 3.1.4 for details. Reading the value of the index of the physical system can be obtained from these registers. Writing to the registers will clear the index status flags in the status register.

Address: BASE + 0x06h / BASE + 0x08h / BASE + 0x0Ah

Attribute:read

Data Format:

Bit	7	6	5	4	3	2	1	0
BASE+6/8/A	CB7	CB6	CB5	CB4	CB3	CB2	CB1	CB0
BASE+7/9/B	CB15	CB14	CB13	CB12	CB11	CB10	CB9	CB8

CB15…CB0: Counter value.

CB15 is the Most Significant Bit (MSB)

CB0 is the Least Significant Bit (LSB).

Address: BASE + 0x06h / BASE + 0x08h / BASE + 0x0Ah

Attribute: write

Data Format:

Bit	7	6	5	4	3	2	1	0
BASE+6/8/A	X	X	X	X	X	X	X	X
BASE+7/9/B	X	X	X	X	X	X	X	X

4.4 Status Register

The register shows the status of the three general purpose differential input signals and the index latch registers.

Address: BASE + 0x0Ch

Attribute: read only

Data Format:

Bit	7	6	5	4	3	2	1	0
BASE+ 0x0C	1	IDL1	IDL2	IDL3	1	IN2	IN1	IN0
BASE+ 0x0D	0	0	0	0	0	0	0	DIR

IN2~IN0: The inputs from 3 differential photo-isolated pins.

IDL3~IDL1: The status of the index latch register. The initial values of these bits are zero. The status bits are set to 1 at the rising edge of the index signals (PHC3~PHC1), and are reset to zero by writing to the index register of the respective channels.

DIR: This bit shows the counting direction of the counter 3, DIR="1" meaning up counting.

4.5 Digital Output Register

This register sets the isolated digital output pins.

Address: BASE + 0x10h

Attribute: read only

Data Format:

Bit	7	6	5	4	3	2	1	0
BASE+ 0x10	DO7	DO6	DO5	DO4	DO3	DO2	DO1	DO0
BASE+ 0x11	X	X	X	X	X	X	X	X

DO7~DO0: Bit 7~ bit 0 of the isolated digital input.

4.6 Control Mode Register

This register controls the counter operation modes and the PWM output waveform. Counter3 is only for A/B phase mode.

Address: BASE + 0x12h

Attribute: write only

Data Format:

Bit	7	6	5	4	3	2	1	0
BASE+ 0x0E	PE	PCM	X	FTS1	FTS0	C2M1	C2M0	C1MS
BASE+ 0x0F	X	X	X	X	X	X	X	X

C1MS: The mode select bit of counter 1. The input signals of counter 1 are from PHA1, /PHA1, PHB1 and /PHB1.

C1MS	Operation Modes of counter 1
0	Input Signal is (CCW+CW) PHA1: CW (Up count pulse) PHB1: CCW (Down count pulse)
1 (default)	Input Signal is A, B phase encoder

C2M1, C2M0: The mode control bits of counter 2. The input signals of counter 2 are from PHA2, PHA12, PHB2 and PHB2. There are four operating modes for counter 2.

C2M1	C2M0	Operation Modes of counter 2
0	0	PHA2: OUT (Pulse output) PHB2: DIR (Direction)
0	1	PHA2: CW (Up count pulse) PHB2: CCW (Down count pulse)
1	0	A,B phase encoder input (Digital filter not used)
1	1	A, B phase encoder input (Signals pass through digital filter)

FTS1, FTS2: These bits are used to select the time base of the digital filters. Three 3-stage digital filters are used for filtering out noise from the encoder input. Users should set the values according to the operating conditions. For example, while (FTS1, FTS0) = (1, 0), the signal level transition period, which is less than 1.2 μs, will be filtered and ignored for the counter. The setting of the time base is:

FTS1	FTS0	Digital Filter Time Period
0	0	300 ns
0	1	600 ns
1	0	1.2 μs
1	1	4.8 μs

PCM: PWM Control Mode bit. Its default is 0 for symmetric PWM signals and 1 is for non-symmetric PWM signals. A real-time system is needed to use mode 1. The user needs to set the other half pulse width every time when INT0 is coming. Otherwise, users can use mode 0 for symmetric pulse width output.

PE: PWM output enabled control. Six PWM waveforms output are enabled when this bit is set to 1, otherwise they are all zero voltage output (PE=0). Its default value is 0. This bit also controls the INT0 interrupt output signal. PE must be set to 1 to enable the INT0 interrupt source.

4.7 Interrupt #0 Period Register

This register is used to set the period to generate INT0 interrupt signals. It is a 12-bit register. Refer to section 3.8 for operation details.

Address: BASE + 0x16h

Attribute: write only

Data Format:

Bit	7	6	5	4	3	2	1	0
BASE+0x16	CV7	CV6	CV5	CV4	CV3	CV2	CV1	CV0
BASE+0x17	X	X	X	X	CV11	CV10	CV9	CV8

CV11...CV0: Timer value.

CV11 is the MSB

CV0 is the LSB.

4.8 Interrupt #1 Period Register

This register is used to set the period to generate INT1 interrupt signals and the dead time of the PWM output signals. The register is divided into two parts: the upper byte is used to set the INT1 period and the lower byte is used to set the PWM dead time.

Refer to Chapter 3 for details on how to control the INT1 period and PWM dead time.

Address: BASE + 0x18h

Attribute: write only

Data Format:

Bit	7	6	5	4	3	2	1	0
BASE+0x18	N7	N6	N5	N4	N3	N2	N1	N0
BASE+0x19	M7	M6	M5	M4	M3	M2	M1	M0

N7~N0: PWM Dead Time control value

M7~M0: INT1 control value

4.9 PWM Output Registers

There are three PWM output channels on the PCI-8133. Values in the registers are used to set the “High” period of the PWM signals: (UU, UD), (VU, VD), (WU, WD). The time period has 12 bits resolution. Refer to section 3.7 for details of how to set the PWM.

Address: BASE + 0x1Ah / +0x1Ch / 0x1Eh

Attribute: write only

Data Format:

Bit	7	6	5	4	3	2	1	0
BASE+0x1A	PM7	PM6	PM6	PM4	PM3	PM2	PM1	PM0
BASE+0x1B	X	X	X	X	PM11	PM10	PM9	PM8

4.10 Digital Input Register

This register indicates the value of the isolated digital input channels. There are 8 DI channels.

Address: BASE + 0x40h

Attribute: read only

Data Format:

Bit	7	6	5	4	3	2	1	0
BASE+0x40	DI7	DI6	DI5	DI4	DI3	DI2	DI1	DI0
BASE+0x41	X	X	X	X	X	X	X	X

4.11 Interrupt Clear Registers

There are two interrupt clear registers. After processing an interrupt by means of software ISR, these registers must be written to clear the interrupt flags, so that it can enable the next interrupt trigger. Addresses 0x80h and 0x90h are used to clear INT0 and INT1 respectively.

Address: BASE + 0x80h / BASE + 0x90h

Attribute: write only

Data Format:

Bit	7	6	5	4	3	2	1	0
BASE+0x80	X	X	X	X	X	X	X	X
BASE+0x90	X	X	X	X	X	X	X	X

5

C/C++ Library

5.1 Installation

5.1.1 MS-DOS Software Installation

We have provided sample programs and Borland C++ 3.1 static link libraries for this card. You can locate these materials in the following path

X:\Motion_Control\PCI-8133\DOS_BC

5.1.2 Device Installation for Windows 95/98/NT/2000

Run the `setup.exe` file from the ADLINK all-in-one CD and choose the Driver Installation option. Select the `/Motion Control/PCI-8133` directory, select the OS, and follow the setup instructions to complete the installation. A reboot dialog box will appear. Power off your computer, plug the PCI-8133 card into one of the PCI slots and power on the computer.

After powering back up, the system will automatically detect the PCI-8133 card and display a dialog box that will prompt you to select the device information. Click on ***"Next step"*** and the system will find the PCI-8133.

5.2 C/C++ Programming Library

This section provides detailed information of all functions. Function prototypes and some common data types are declared in PCI-8133.h. We suggest you use these data types in your application programs. The following table shows the data type names and their range.

Data Types

Type Name	Description	Range
U8	8-bit ASCII character	0 to 255
I16	16-bit signed integer	-32768 to 32767
U16	16-bit unsigned integer	0 to 65535
I32	32-bit signed integer	-2147483648 to 2147483647
U32	32-bit single-precision floating-point	0 to 4294967295
F32	32-bit single-precision floating-point	-3.402823E38 to 3.402823E38
F64	64-bit double-precision floating-point	-1.797683134862315E308 to 1.797683134862315E309
Boolean	Boolean logic value	TRUE, FALSE

The functions of the PCI-8133's software drivers use full names to represent the functions' real meaning. The naming convention rules are:

Under a DOS Environment:

_{hardware_model}_{action_name}. E.g. **`_8133_Initial()`**.

In order to recognize the difference between a DOS library and a Windows 95/98/NT library, A capital "**W**" is place at the head of each function name for Windows 95/98/NT DLL driver (e.g. **`W_8133_Initial()`**).

Descriptions of each function are specified in the proceeding sections.

5.3 _8133_Initial

@ *Description*

This function is used to initialize the PCI-8133 card. Each PCI-8133 card must be initialized with this function before calling any other function.

@ *Syntax*

C/C++ (DOS)

```
U16 _8133_Initial (U16 *existCards, PCI_INFO *info)
```

C/C++ (Windows 95/98)

```
U16 W_8133_Initial (U16 *existCards, PCI_INFO *info)
```

C/C++ (Windows NT/2000)

```
U16 W_8133_Initial (U16 CardNo)
```

Visual Basic (Windows 95/98)

```
W_8133_Initial (existCards As Integer, info As PCI_INFO)
As Integer
```

Visual Basic (Windows NT/2000)

```
W_8133_Initial (ByVal CardNo As Integer) As Integer
```

@ *Argument*

existCards:	number of existing PCI-8133 cards
CardNo:	Assigned card number
info:	information on PCI-8133 cards

@ *Return Code*

```
ERR_NoError
ERR_BoardNoInit
ERR_PCIBiosNotExist
```

5.4 _8133_Software_Reset

@ *Description*

This function is used to reset the I/O port configuration. Note that this function does not reset the PCI bus nor do the hardware settings change. It only resets the register map values to default settings.

@ *Syntax*

C/C++ (DOS)

```
void _8133_Software_Reset (U16 cardNo)
```

C/C++ (Windows 95/98/NT/2000)

```
void W_8133_Software_Reset (U16 cardNo)
```

Visual Basic (Windows 95/98/NT/2000)

```
W_8133_Software_Reset (ByVal cardNo As Integer)
```

@ *Argument*

cardNo: card number

@ *Return Code*

```
ERR_NoError
ERR_BoardNoInit
ERR_PCIBiosNotExist
```

5.5 _8133_Read_Cnt

@ *Description*

This function is used to read data from the 3 independent unsigned 16-bit up/down counters, which range from 0h to 0xFFFFh. The operating modes of these 3 counters are defined by the 8133_ModeSelect function.

@ *Syntax*

C/C++ (DOS)

```
U16 _8133_Read_Cnt (U16 cardNo, U16 CntNo, U16 *CntData)
```

C/C++ (Windows 95/98/NT/2000)

```
U16 W_8133_Read_Cnt (U16 cardNo, U16 CntNo, U16 *CntData)
```

Visual Basic (Windows 95/98/NT/2000)

```
W_8133_Read_Cnt (ByVal cardNo As Integer, ByVal CntNo As
Integer, CntData As Integer) As Integer
```

@ *Argument*

cardNo: card number
CntNo: Counter Number (= 1, 2, 3)
CntData: Data read from counter (= 0h .. FFFFh)

@ *Return Code*

ERR_BoardNoInit
ERR_NoError

5.6 _8133_Read_Index

@ Description

This function is used to read data from the 3 index registers associated with the three up/down counters. The value of the up/down counters (Counter No.=1, 2, 3) are latched onto the index registers (Index No.=1, 2, 3) on the rising edge of its associated index signal (PHC1, PHC2, and PHC3). The latching status of each index register can be read using the _8133_Read_Status function.

@ Syntax

C/C++ (DOS)

```
U16  _8133_Read_Index (U16 cardNo, U16 IndexNo, U16
*IndexData)
```

C/C++ (Windows 95/98/NT/2000)

```
U16  W_8133_Read_Index (U16 cardNo, U16 IndexNo, U16
*IndexData)
```

Visual Basic (Windows 95/98/NT/2000)

```
W_8133_Read_Index (ByVal cardNo As Integer, ByVal IndexNo
As Integer, IndexData As Integer) As Integer
```

@ Argument

cardNo: card number
IndexNo: Index Number (= 1, 2, 3)
IndexData: Index Value (= 0H .. FFFFh)

@ Return Code

ERR_BoardNoInit
ERR_NoError

5.7 _8133_Read_Status

@ Description

This function is used to read data from the status register. The definition of each bit in this register is described in section 4.4

@ Syntax

C/C++ (DOS)

```
U16  _8133_Read_Status (U16 cardNo, U16 *Status)
```

C/C++ (Windows 95/98/NT/2000)

```
U16  W_8133_Read_Status (U16 cardNo, U16 *Status)
```

Visual Basic (Windows 95/98/NT/2000)

```
W_8133_Read_Status (ByVal cardNo As Integer, Status As
Integer) As Integer
```

@ Argument

cardNo: card number
Status: Value in Status Reg.

@ Return Code

ERR_BoardNoInit, ERR_NoError

5.8 _8133_CLR_IdxLah

@ *Description*

This function resets the latching status of the index register.

@ *Syntax*

C/C++ (DOS)

```
U16  _8133_CLR_IdxLah (U16 cardNo, U16 IndexNo)
```

C/C++ (Windows 95/98/NT/2000)

```
U16  W_8133_CLR_IdxLah (U16 cardNo, U16 IndexNo)
```

Visual Basic (Windows 95/98/NT/2000)

```
W_8133_ CLR_IdxLah (ByVal cardNo As Integer, ByVal
IndexNo As Integer) As Integer
```

@ *Argument*

cardNo:	card number
IndexNo:	The index number for the latching bit to be cleared

@ *Return Code*

```
ERR_BoardNoInit
ERR_NoError
```

5.9 _8133_ModeSelecct

@ Description

This function is used to set the control mode register. The definition of each bit in the control word is described in section 4.6.

@ Syntax

C/C++ (DOS)

```
U16  _8133_ModeSelect (U16 cardNo, U16 Mode)
```

C/C++ (Windows 95/98/NT/2000)

```
U16  W_8133_ModeSelect (U16 cardNo, U16 Mode)
```

Visual Basic (Windows 95/98/NT/2000)

```
W_8133_ModeSelect (ByVal cardNo As Integer, ByVal Mode As
Integer) As Integer
```

@ Argument

cardNo: card number
Mode: Control word to be written

@ Return Code

ERR_BoardNoInit
ERR_NoError

5.10 _8133_Set_Int0Perd

@ *Description*

This function is used to write to the INT0 period control registers. Refer to Section 4.7 for details of these registers.

@ *Syntax*

C/C++ (DOS)

```
U16  _8133_Set_Int0Perd (U16 cardNo, U16 Int0Perd)
```

C/C++ (Windows 95/98/NT/2000)

```
U16  W_8133_Set_Int0Perd (U16 cardNo, U16 Int0Perd)
```

Visual Basic (Windows 95/98/NT/2000)

```
W_8133_Set_Int0Perd (ByVal cardNo As Integer, ByVal
Int0Perd As Integer) As Integer
```

@ *Argument*

cardNo: card number
Int0Perd: Interrupt 0 period to be set

@ *Return Code*

ERR_BoardNoInit
ERR_NoError

5.11 _8133_Set_Int1Perd

@ *Description*

This function is used to set the INT1 interrupt period and PWM dead time of the control register. Refer to section 5.11 for details.

@ *Syntax*

C/C++ (DOS)

```
U16  _8133_Set_Int1Perd (U16 cardNo, U16 Int1Perd)
```

C/C++ (Windows 95/98/NT/2000)

```
U16  W_8133_Set_Int1Perd (U16 cardNo, U16 Int1Perd)
```

Visual Basic (Windows 95/98/NT/2000)

```
W_8133_Set_Int1Perd (ByVal cardNo As Integer, ByVal
Int1Perd As Integer) As Integer
```

@ *Argument*

cardNo: card number
Int1Perd: Interrupt period to be set

@ *Return Code*

ERR_BoardNoInit
ERR_NoError

5.12 _8133_Set_PWMPerd

@ *Description*

This function is used to set the period for generating the three complementary PWM signals (UU, UD), (VU, VD), and (WU, WD). Refer to Section 4.9 for details of the PWM control.

@ *Syntax*

C/C++ (DOS)

```
U16  _8133_Set_PWMPerd (U16 cardNo, U16 PWMNo, U16
PWMPerd)
```

C/C++ (Windows 95/98/NT/2000)

```
U16  W_8133_Set_PWMPerd (U16 cardNo, U16 PWMNo, U16
PWMPerd)
```

Visual Basic (Windows 95/98/NT/2000)

```
W_8133_Set_PWMPerd (ByVal cardNo As Integer, ByVal PWMNo
As Integer, ByVal PWMPerd As Integer) As Integer
```

@ *Argument*

cardNo:	card number
PWMNo:	PWMNo = 1, 2, 3 for U, V, W arm respectively
PWMPerd:	PWM period to be set

@ *Return Code*

```
ERR_BoardNoInit
ERR_NoError
```

5.13 _8133_INT_Control

@ Description

The PCI-8133 uses a dual interrupt system. The dual interrupt sources can be generated and be checked by the software. This function is used to select and control the PCI-8133 interrupt sources by writing data to the interrupt control register.

@ Syntax

C/C++ (DOS)

```
void _8133_INT_Control (U16 cardNo, U16 int0Ctrl , U16
int1Ctrl)
```

C/C++ (Windows 95/98/NT/2000)

```
void W_8133_INT_Control (U16 cardNo, U16 int0Ctrl , U16
int1Ctrl)
```

Visual Basic (Windows 95/98/NT/2000)

```
W_8133_INT_Control (ByVal cardNo As Integer, ByVal
int0Ctrl As Integer, ByVal int1Ctrl As Integer) As Integer
```

@ Argument

```
cardNo:  the card number of PCI-8133 card initialized.
Int0Ctrl:0: INT0 disable
         1: INT0 enable
int1Ctrl:0:INT1 disable
         1: INT1 enable
```

@ Return Code

None

5.14 _8133_CLR_IRQ0 & _8133_CLR_IRQ1

@ Description

These functions are used to clear interrupt requests generated by the PCI-8133.

@ Syntax

C/C++ (DOS)

```
void _8133_CLR_IRQ0 (U16 cardNo)
void _8133_CLR_IRQ1 (U16 cardNo)
```

C/C++ (Windows 95/98/NT/2000)

```
void W_8133_CLR_IRQ0 (U16 cardNo)
void W_8133_CLR_IRQ1 (U16 cardNo)
```

Visual Basic (Windows 95/98/NT/2000)

```
W_8133_CLR_IRQ0 (ByVal cardNo As Integer)
W_8133_CLR_IRQ1 (ByVal cardNo As Integer)
```

@ Argument

cardNo: card number the of PCI-8133 card initialized.

@ Return Code

None

5.15 _8133_Get_IRQ_Channel

@ Description

This function is used to retrieve the IRQ level of the PCI-8133 card currently being used.

@ Syntax

C/C++ (DOS)

```
void _8133_Get_IRQ_Channel (U16 cardNo, U16 *irq_no )
```

C/C++ (Windows 95/98/NT/2000)

```
void W_8133_Get_IRQ_Channel (U16 cardNo, U16 *irq_no )
```

Visual Basic (Windows 95/98/NT/2000)

```
W_8133_Get_IRQ_Channel (ByVal cardNo As Integer, irq_no
As Integer)
```

@ Argument

cardNo: the card number of PCI-8133 card initialized.
Irq_no: the IRQ level used to transfer A/D data for the card

@ Return Code

None

5.16 _8133_Get_IRQ_Status

@ Description

The PCI-8133 has a dual interrupt system. Two interrupt sources can be generated and checked by software. If both INT1 and INT2 are enabled, this function is then used to distinguish between interrupts.

@ Syntax

C/C++ (DOS)

```
void  _8133_Get_IRQ_Status (U16 cardNo, U16 *ch1, U16
*ch2 )
```

C/C++ (Windows 95/98/NT/2000)

```
void  W_8133_Get_IRQ_Status (U16 cardNo, U16 *ch1, U16
*ch2 )
```

Visual Basic (Windows 95/98/NT/2000)

```
W_8133_Get_IRQ_Status (ByVal cardNo As Integer, ch1 As
Integer, ch2 As Integer)
```

@ Argument

cardNo: card number of the PCI-8133 card initialized.
ch1: the IRQ status of INT 0: interupt is not from INT1, 1: interupt is from INT1 .
ch2: the IRQ status of INT2, 0: interupt is not from INT2, 1: interupt is from INT2 .

@ Return Code

None

5.17 _8133_DO

@ *Description*

This function is used to write data to the digital output port. There are 8 digital output channels supported by the PCI_8133.

@ *Syntax*

C/C++ (DOS)

```
U16  _8133_DO (U16 cardNo, U16 DOData)
```

C/C++ (Windows 95/98/NT/2000)

```
U16  W_8133_DO (U16 cardNo, U16 DOData)
```

Visual Basic (Windows 95/98/NT/2000)

```
W_8133_DO (ByVal cardNo As Integer, ByVal DOData As
Integer) As Integer
```

@ *Argument*

cardNo: card number of the PCI-8133 card initialized
DOData: value to be written to digital output port

@ *Return Code*

ERR_NoError

5.18 _8133_DI

@ *Description*

This function is used to read data from the digital input ports. There are 8 digital input channels supported by the PCI_8133. The digital input status can be accessed using this function.

@ *Syntax*

C/C++ (DOS)

```
U16  _8133_DI (U16 cardNo, U16 *DIData)
```

C/C++ (Windows 95/98/NT/2000)

```
U16  W_8133_DI (U16 cardNo, U16 *DIData)
```

Visual Basic (Windows 95/98/NT/2000)

```
W_8133_DI (ByVal cardNo As Integer, DIData As Integer) As
Integer
```

@ *Argument*

cardNo: card number of the PCI-8133 card initialized
DIData: value accessed from the digital input port

@ *Return Code*

ERR_NoError

5.19 _8133_INT_Enable

@ *Description*

This function is only available in Windows 95/98/NT/2000 drivers. It is used to start up the interrupt control. After calling this function, each time an interrupt request signal is generated, a software event is signaled. The application program can therefore utilize the wait operation function WaitForSingleObject() API and wait for the event. When the event is signaled, an interrupt is generated.

@ *Syntax*

C/C++ (Windows 95/98/NT/2000)

```
U16  W_8133_INT_Enable (U16 cardNo, HANDLE *hEvent)
```

Visual Basic (Windows 95/98/NT/2000)

```
W_8133_INT_Enable (ByVal cardNo As Integer, hEvent As
Long) As Integer
```

@ *Argument*

cardNo:	card number of the PCI-8133 card initialized
hEvent:	Address of an array of two handles. hEvent[0] and Event[1] are the events for interrupt signals INT1 and INT2 respectively.

@ *Return Code*

ERR_NoError

5.20 _8133_INT_Disable

@ *Description*

This function is only available in Windows 95/98/NT/2000 drivers. It is used to disable the generation of interrupt signals.

@ *Syntax*

C/C++ (Windows 95/98/NT/2000)

```
U16  W_8133_INT_Disable (U16 cardNo)
```

Visual Basic (Windows 95/98/NT/2000)

```
W_8133_INT_Disable (ByVal cardNo As Integer) As Integer
```

@ *Argument*

cardNo: card number of the PCI-8133 card initialized

@ *Return Code*

ERR_NoError

A

DOS Example Programs

This appendix contains a demo program to illustrate how to write an ISR for speed and position monitoring under a DOS environment.

```
#define         IC8259_1        0x20
#define         IC8259_2        0xA0
#define         EOI             0x20
void interrupt far   isr(void);
void (interrupt far *old_isr)();
U16     irq_mask,old_mask;
U16     ISR_irqchn;
int     dPLS;
long    Position=0;
U16     pls_N=0, pls_NM1=0;

void main(void)
{
        U16     vect_no, irq_chn;

        _8133_Software_Reset( cno  );
        delay( 1 );

        /*  Set Control mode for index mode, CNT1 & CNT2 at CW/CCW mode
*/
        _8133_ModeSelect( cno, 0x42);

        /*  Set Period of INT0 is 200us */
        _8133_Set_Int0Perd( cno, 0x07d0);

        /*  Set Period of INT1 is 5ms, Dead time 3.5 us */
        _8133_Set_Int1Perd( cno, 0x1803);

        /*  Enable only INT1 */
        _8133_Set_INT_Control( cno, 0, 1 );

        _8133_Get_IRQ_Channel( cno, &irq_chn );  /* Get IRQ Channel
*/
        ISR_irqchn = irq_chn;
```

```
        if ( irq_chn==0)   return 1;
        if ( irq_chn < 8)          vect_no = 0x08 + irq_chn ;
        else if (irq_chn == 9)     vect_no = 0x0A ;
        else                       vect_no = 0x70 + irq_chn - 8;

        disable();
        old_isr = getvect( vect_no  );
        setvect( vect_no  , isr );
        enable();
                                                    /* Enable ISR
*/
        if( irq_chn < 8 )
        {
                irq_mask = inp( IC8259_1 + 1 );
                old_mask = irq_mask ;
                outp( IC8259_1 + 1 , irq_mask &
(0xFF^(1<<irq_chn)) );
        }
        else{
                irq_mask = inp( IC8259_1 + 1 );
                outp( IC8259_1 + 1 , irq_mask & 0xFB );
                                                  /* IRQ2 : 1111
1011 */
                irq_mask = inp( IC8259_2 + 1 );
                old_mask = irq_mask ;
                outp( IC8259_2 + 1 , irq_mask &
(0xFF^(1<<(irq_chn-8) )) );
        }

        /*  MAIN  PROGRAM  */
        if( irq_chn <8 )   outp( IC8259_1 + 1 , old_mask );
        else               outp( IC8259_2 + 1 , old_mask );
        setvect( vect_no  , old_isr );
        else               outp( IC8259_2 + 1 , old_mask );
        setvect( vect_no  , old_isr );
}

void interrupt far isr(void)
{
        _8133_Read_Cnt( cno, 3, &pls_N);
        _8133_CLR_IRQ1(  cno  );
                                   // Clear INT1 Request
        dPLS = (int)pls_N - (int)pls_NM1;
                                   // dPLS(n) = pls(n)-pls(n-1)
                                         // Speed = K x dPLS(n)
        Position += dPLS;             // P(n) = P(n-1) + dPLS(n)
        pls_NM1 = pls_N;              // update pls(n-1)

        if( ISR_irqchn > 8 )  outp( IC8259_2, EOI );
        outp( IC8259_1 , EOI );
}
```

B

PWM Example

The outputs from the PWM pins are open collectors and can have a sink current of up to 20mA. External pull up resistors is necessary for PWM wiring.

Hardware: Pin wiring

(U+) Pin 16 cascades a 10k resister to Pin 19 （VCC）

(U-) Pin 35 cascades a 10k resister to Pin 19 （VCC）

(OENA) Pin 34 connects to Pin 19（VCC）. This will enable the PWM.

Software: Function

_8133_set_PWMPerd（） to set bandwidth.

_8133_set_INT0Perd（） to set frequency.

_8133_ModeSelecct（） to set the control mode resister（0X82）.

_8133_set_INT1Perd（...&Hxxyy） yy is dead time.

Dead time range ➔ 3μs < dead time<192 μs

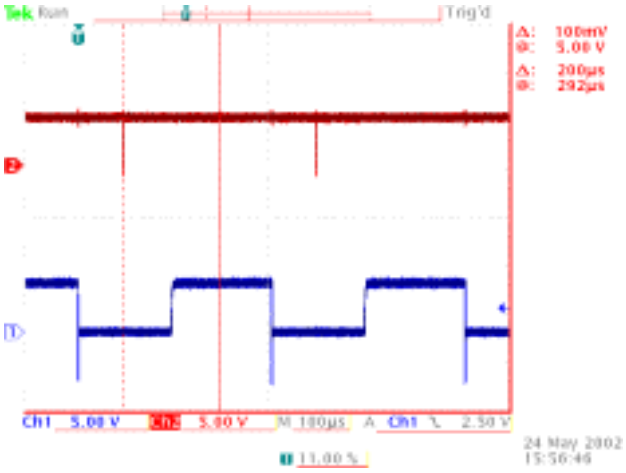
Users can set any PWM duty cycle time. After setting the dead time, they need to be collocated if the user needs different duty cycle times.

- Channel 1=PWM (U-) Channel 2=INT0 (pin E201.80)

  Control Mode Register PCM=0

  PWMPried=1000, INT0=2000

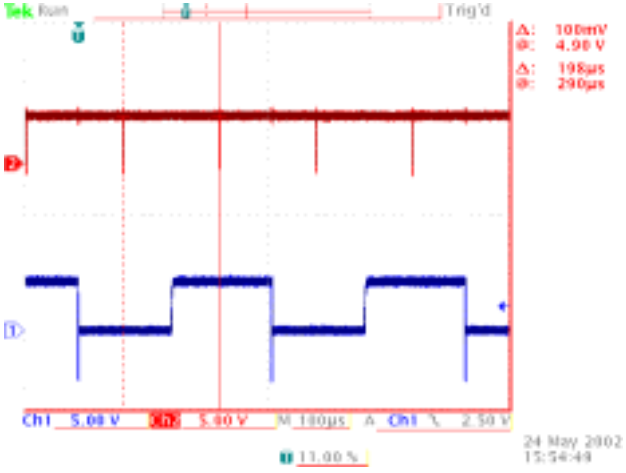
  Result PWMPeriod = 400us, INT0 = 400us



Control Mode Register PCM=1

Set PWMPeriod=1000, INT0=2000

Result PWMPeriod =400us, INT0=200us



C

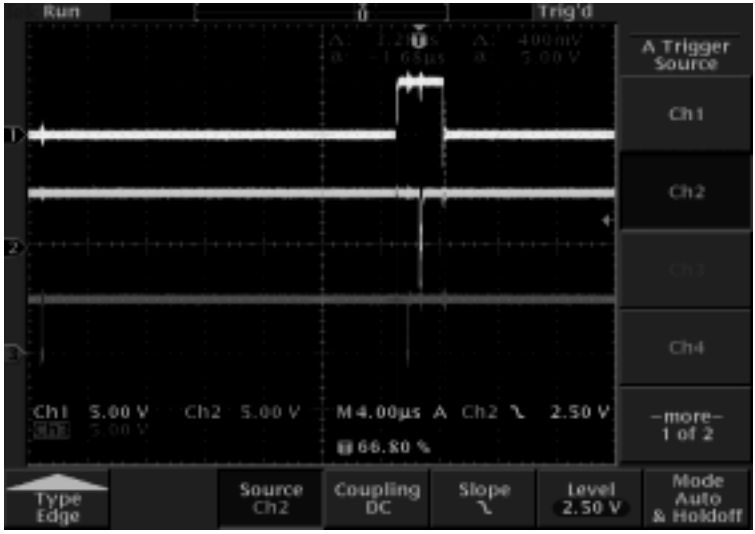
PWM Duty Cycle Example

- INT1=100xINT0=20ms Int1<255
- Dead Time 1.5us ~ 192us TDT=0.75*(m+1)us (0<m<255)
  W_8133_Set_Int1Perd (0,0x6301); m=1 Int1=100
- Enable PWM mode
  W_8133_ModeSelect(0, 0xC2);
- Period 9 ~ 242
  W_8133_Set_PWMPerd(0, 1, 242)

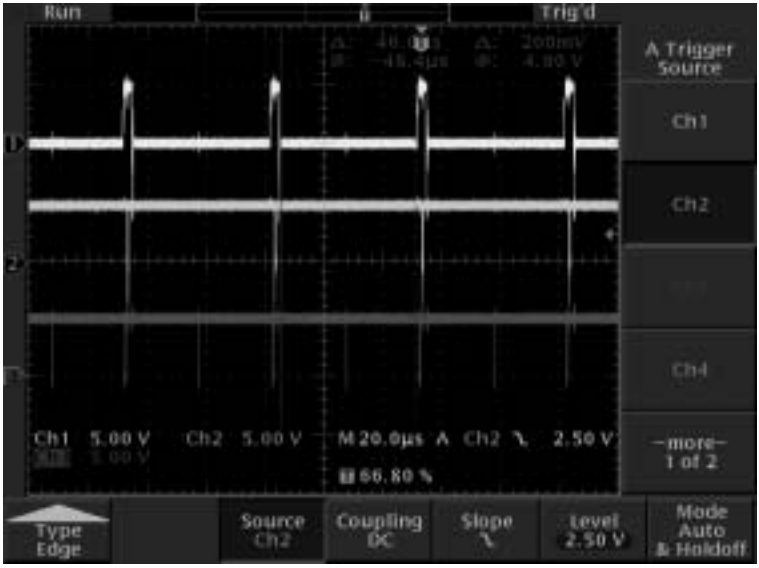
The following diagram shows the test result of a PWM period = 242

Channel1=U+ Channel2 is=U- Channel3=INT0:

PCM=0


Run
Trig'd
A Trigger Source
Ch1
Ch2
Ch4
Ch1 5.00 V
Ch2 5.00 V
M4.00µs A Ch2 2.50 V
66.80 %
–more–
1 of 2
Type Edge
Source Ch2
Coupling DC
Slope
Level 2.50 V
Mode Auto & Holdoff

PCM=1


Run
Trig'd
A Trigger Source
Ch1
Ch2
Ch4
Ch1 5.00 V
Ch2 5.00 V
M20.0µs A Ch2 2.50 V
66.80 %
–more–
1 of 2
Type Edge
Source Ch2
Coupling DC
Slope
Level 2.50 V
Mode Auto & Holdoff

- INT1=100xINT0=20ms Int1<255
- Dead Time 1.5us ~ 192us TDT=0.75*(m+1) us (0<m<255)

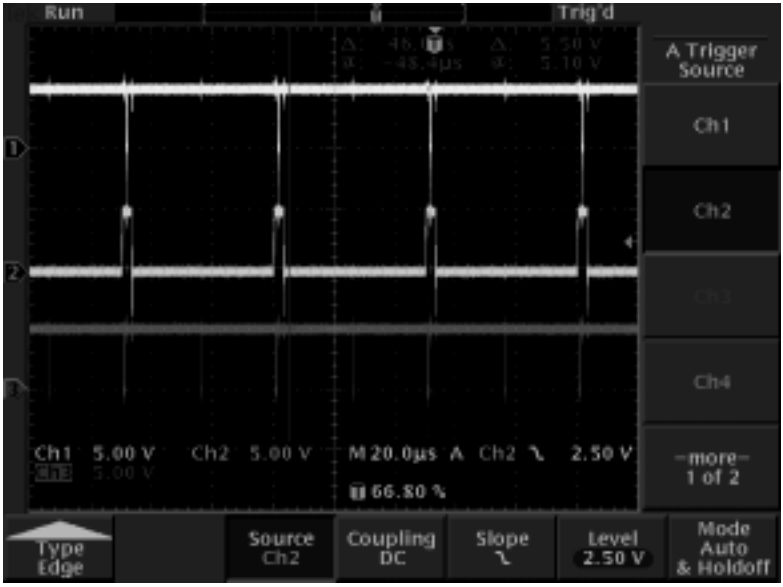
  W_8133_Set_Int1Perd(0, 0x6301) m=1, Int1=100
- Enable PWM mode

  W_8133_ModeSelect(0, 0xC2)
- Period 9 ~ 242

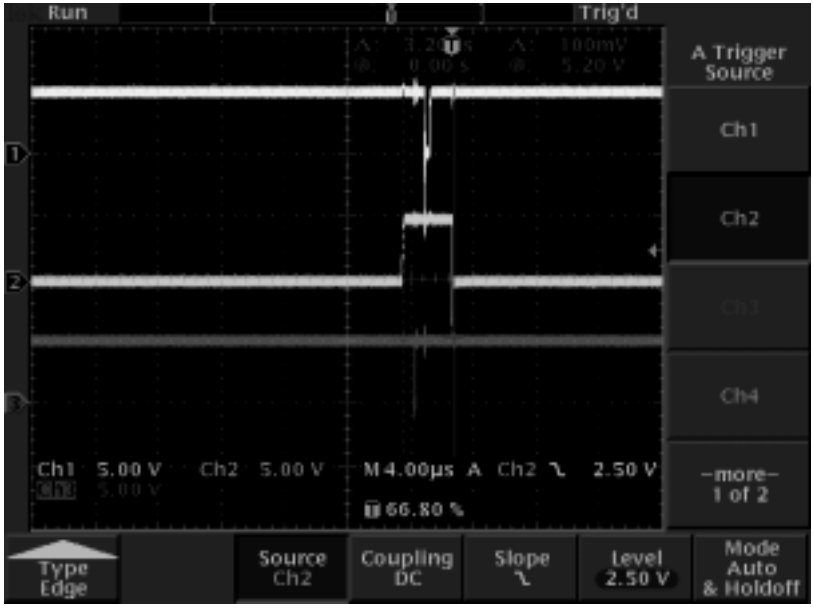
  W_8133_Set_PWMPerd (0, 1, 9)

The following diagram shows the test result of PWM period = 9, Channel1 is U+, Channel2 is U-, and Channel3 is INT0:

PCM=0



PCM=1


Run
Trig'd
A Trigger
Source
Ch1
Ch2
Ch3
Ch4
Ch1 5.00 V
Ch2 5.00 V
M4.00µs A Ch2 2.50 V
66.80 %
-more-
1 of 2
Type
Edge
Source
Ch2
Coupling
DC
Slope
Level
2.50 V
Mode
Auto
& Holdoff

Warranty Policy

Thank you for choosing ADLINK. To understand your rights and enjoy all the after-sales services we offer, please read the following carefully.

1. Before using ADLINK's products please read the user manual and follow the instructions exactly. When sending in damaged products for repair, please attach an RMA application form.

2. All ADLINK products come with a two-year guarantee, repaired free of charge.

    - The warranty period starts from the product's shipment date from ADLINK's factory
    - Peripherals and third-party products not manufactured by ADLINK will be covered by the original manufacturers' warranty
    - End users requiring maintenance services should contact their local dealers. Local warranty conditions will depend on the local dealers.

3. Our repair service does not cover the two-year warranty, if the following items cause damage:

    a. Damage caused by not following instructions on user menus.

    b. Damage caused by carelessness on the users' part during product transportation.

    c. Damage caused by fire, earthquakes, floods, lightening, pollution, and/or incorrect usage of voltage transformers.

    d. Damage caused by unsuitable storage environments (i.e. high temperatures, high humidity or volatile chemicals.

    e. Damage caused by leakage of battery fluid when changing batteries.

    f. Damage from improper repair by unauthorized technicians.

    g. Products with altered and/or damaged serial numbers are not entitled to our service.

    h. Other categories not protected under our guarantees.

4. Customers are responsible for shipping costs to transport damaged products to our company or sales office.

5. To ensure the speed and quality of product repair, please download a RMA application form from our company website www.adlinktech.com. Damaged products with RMA forms attached receive priority.

For further questions, please contact our FAE staff.

ADLINK: service@adlinktech.com

Test & Measurement Product Segment: NuDAQ@adlinktech.com

Automation Product Segment: Automation@adlinktech.com

Computer & Communication Product Segment: NuPRO@adlinktech.com; NuIPC@adlinktech.com